

فشرده‌سازی اطلاعات متغیر با زمان با استفاده از کد هافمن

حسن فرسی^۱ و پوریا اعتضادی فر^۲

^۱ دانشکده مهندسی برق و کامپیوتر، دانشگاه بیرجند، بیرجند، ایران

چکیده

در این مقاله با استفاده از شبکه عصبی مصنوعی، تابعی بر منحنی چگالی احتمال، رشته‌اطلاعات افراز می‌شود. این امر باعث می‌شود که منحنی چگالی احتمال رشته‌اطلاعات به وسیله تعدادی پارامتر تقریب زده شده و از آن به منظور حافظه‌دار کردن منحنی چگالی احتمال و نیز جهت فشرده‌سازی اطلاعات با روش هافمن استفاده می‌شود. تفاوت بین روش پیشنهادی و روش عمومی الگوریتم هافمن در این است که در روش پیشنهادی به جهت حافظه‌دار کردن منحنی چگالی احتمال، می‌توان چندین بار از الگوریتم هافمن برای کد کردن رشته‌اطلاعات استفاده کرد و در هر بار کد کردن اطلاعات، تابع چگالی احتمال را تقریب زده و اطلاعات مربوط به تابع چگالی احتمال را به انتهای رشته مورد نظر اضافه کرد. بنابراین علی‌رغم متغیر بودن تابع چگالی احتمال در هر مرحله، الگوریتم هافمن با استفاده از روش پیشنهادی قابل پیاده‌سازی است. همچنین دو الگوریتم برای کد کردن و دیکد کردن رشته‌اطلاعات پیشنهاد شده است. در انتها درصد فشرده سازی روش پیشنهادی با دو روش FDR Code و Golomb مقایسه می‌گردد.

واژگان کلیدی: تابع چگالی احتمال متغیر با زمان - شبکه عصبی - فشرده سازی اطلاعات - حافظه‌دار کردن رشته‌اطلاعات - کد هافمن.

۱- مقدمه

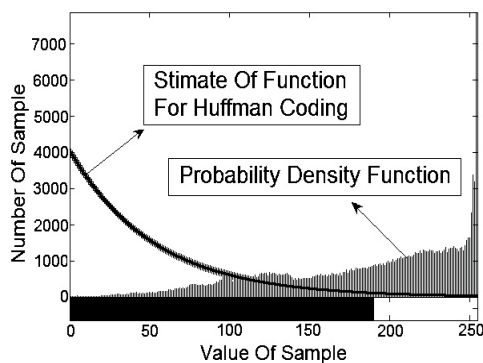
در علوم رایانه و تئوری اطلاعات، الگوریتم هافمن یک الگوریتم کدگذاری برای فشرده‌سازی بی اتلاف اطلاعات است (Huffman, 1952, Proakis, 1995). این تعبیر به استفاده از جدول کد طول متغیر برای کد کردن هر کدام از نشانه‌های مبدأ برمی‌گردد. جدول کد طول متغیر، از روش ویژه‌ای مبتنی بر احتمال وقوع هر کدام از نشانه‌های مبدأ به دست می‌آید. این روش توسط دیوید هافمن توسعه یافت (Proakis, 1995).

در کدگذاری هافمن، از روشی خاص جهت نحوه نمایش هر نماد استفاده می‌شود. به این روش، کدهای پیشوندی گفته می‌شود. منظور از کدهای پیشوندی آن است که در این روش رشته‌ای که نشان دهنده یک نویسه خاص است، هیچ‌گاه پیشوند رشته دیگر که نمایان‌گر نویسه دیگر است، نیست. در این روش نویسه‌های پرکاربردتر با طول

رشته‌بیت کوتاه‌تری نسبت به نویسه‌هایی که کاربردشان کمتر است، نشان داده می‌شوند (Huffman, 1952). بنابراین رشته‌بیت ارسالی با استفاده از این روش کوتاه‌تر از حالتی است که به تمام اطلاعات تعداد بیت برابری اختصاص داده شود.

در بخش ۲ نشان داده می‌شود که کدگذار هافمن توانایی بالایی در میزان فشرده‌سازی رشته‌اطلاعات با تابع چگالی احتمال متغیر با زمان را ندارد. در این مقاله با استفاده از حافظه‌دار کردن تابع چگالی احتمال اطلاعات و شبکه عصبی، می‌توان برای هر رشته تابع چگالی احتمال مربوط به آن رشته را ارسال کرد. همچنین به دلیل توانایی در ذخیره کردن اطلاعات تابع چگالی احتمال، می‌توان الگوریتم فشرده‌سازی را چندین بار تکرار کرده و رشته‌اطلاعات ارسالی را عاری از هر گونه خطا و تلفات بازسازی کرد.

رشته‌هایی که دارای تابع چگالی احتمال متغیر با زمان هستند، نمی‌توان یک تابع ثابت با زمان را به‌عنوان تابع چگالی احتمال رشته‌اطلاعات در نظر گرفت. چرا که اگر یک تابع ثابت با زمان به عنوان تابع چگالی احتمال انتخاب گردد، کد هافمن دیگر قادر به فشرده‌سازی اطلاعات به بهترین نحو نیست. به‌عبارت دیگر پس از گذشت مدت زمانی، تابع چگالی احتمال دچار تغییر شده، در حالی که سامانه با توجه به تابع چگالی احتمال اولیه، عملیات فشرده‌سازی را انجام می‌دهد. بنابراین به‌اشتباه اطلاعاتی که احتمال رخداد آنها کم است، با تعداد بیت کم و اطلاعاتی را که احتمال رخداد آنها زیاد است با تعداد بیت بیش‌تر کد می‌کند. پس از کدگشایی، به‌دلیل اینکه تابع چگالی احتمال به‌درستی انتخاب نشده، عدم فشرده‌سازی بالای اطلاعات را به‌همراه داشته است (Chandra, et al., 2003) و در مواردی حتی رشته کدشده ارسالی دارای تعداد بیت بیش‌تری نسبت به رشته کدشده اصلی است (شکل ۱، جدول ۱).



(شکل ۱): نمونه‌ای از تخمین نادرست تابع چگالی احتمال به دلیل متغیر بودن هیستوگرام تعداد تکرار نمونه‌ها در رشته‌اطلاعات

همان‌طور که در جدول ۱ نشان داده شده است، در صورت بروز خطا در تقریب تابع چگالی احتمال نمونه‌ها، تعداد بیت ساخته‌شده با روش هافمن بیشتر از بیت‌های ورودی است. این مشکل زمانی به‌وجود می‌آید که تابع چگالی احتمال اطلاعات متغیر با زمان باشد. متغیر با زمان بودن تابع چگالی احتمال ممکن است پس از مدتی حالتی مانند شکل ۱ را

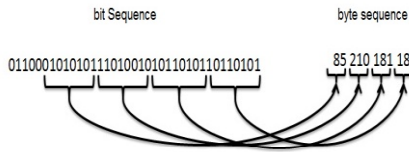
۲ در این مقاله تابع چگالی احتمال مربوط به رشته‌هایی با مقادیر [۲۵۵ و ۳ و ۲ و ۱] که در مبنای دودویی شامل تک‌بایت اطلاعات می‌شوند، می‌باشد. این تابع بر اساس تعداد تکرار این ۲۵۶ عدد در یک رشته‌بایت اطلاعات ساخته می‌شود

در این مقاله کاستی‌های کد هافمن در کد کردن رشته‌هایی با تابع چگالی متغیر با زمان نشان داده می‌شوند. در ادامه روش پیشنهادی گام به گام توضیح داده می‌شود. در بخش سه نحوه به‌دست‌آوردن تابع چگالی احتمال برای یک رشته‌بیت با طول دلخواه بررسی و سپس در بخش چهار مفهوم شبکه عصبی به‌صورت مختصر بیان می‌شود. در بخش پنج منظور از حافظه‌دار کردن رشته‌اطلاعات را توضیح داده و سپس به مزیت حافظه‌دار کردن رشته‌اطلاعات پرداخته می‌شود. در ادامه به توضیح الگوریتم فشرده‌سازی به‌همراه یک مثال برای واضح‌تر شدن مسأله پرداخته می‌شود. در قسمت پایانی، برای ارزیابی الگوریتم پیشنهادی، درصد فشرده‌سازی روش پیشنهادی با روش‌های FDR (Chandra, et al., 2003) و Code (Chandra, et al., 2000, Chakrabarty, et al., 2001, Iyengar, et al., 1999, Jas, et al., 1999, Touba, et al., 1998) مقایسه می‌شود.

۲- مشکل کد هافمن

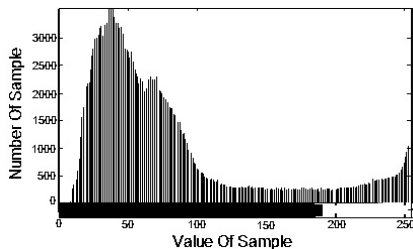
عملکرد کد هافمن (Huffman, 1952, Proakis, 1995) بر پایه احتمال رخداد رشته‌هاست. در این روش رشته‌هایی که احتمال رخداد بیشتری دارند با بیت‌های کمتری ارسال می‌شوند، اما این روش کدگذاری محدودیت دارد. محدودیت کدگذاری با روش هافمن این است که باید قبل از ارسال اطلاعات، تابع چگالی احتمال رخداد نمونه‌ها^۲، برای داده‌هایی که قرار است ارسال شوند، محاسبه شود. همچنین تابع چگالی احتمال باید در گیرنده نیز ساخته شده باشد، یا اینکه هم‌زمان با داده‌های ارسالی به گیرنده ارسال شود، تا گیرنده بتواند داده‌های دریافتی را به‌درستی بازبایی کند. اما در مواقعی که تابع چگالی احتمال رخداد نمونه‌ها متغیر با زمان است دیگر کدگذاری به روش هافمن، به‌تنهایی، کارایی مناسبی ندارد؛ زیرا در ابتدا تابع چگالی احتمالی به‌صورت میانگین تمام اطلاعات ارسالی فرض می‌شود و یا اینکه تابع چگالی احتمال برای اولین رشته تخمین زده شده و بر اساس آن سیستم شروع به کدگذاری و کدگشایی می‌کند. اما برای

^۱ نتایج گزارش شده برای درصد فشرده‌سازی از (Chandra, 2000) و (Chakrabarty, 2001) بهتر است به‌دلیل استفاده از روش بهبود الگوریتم غیر مستدل دوباره مرتب‌کردن



(شکل ۲): نحوه جدا کردن بیت‌ها از رشته‌اطلاعات و ساختن رشته‌ای از بایت‌ها

با این کار بافر به جای اعداد ۱ و ۰ اعدادی بین ۰ تا ۲۵۵ را در مبنای ده شامل می‌شود. در مرحله بعد، تعداد تکرار این ۲۵۶ عدد در رشته مورد نظر محاسبه می‌شود. بنابراین می‌توان هیستوگرام را بر حسب تعداد تکرار این ۲۵۶ عدد ساخته شده در رشته مورد نظر به دست آورد. به عنوان مثال، شکل ۳ نمونه‌ای از هیستوگرام یک رشته‌بیت پس از تبدیل را نشان می‌دهد.



(شکل ۳): هیستوگرام تعداد تکرار نمونه‌ها در یک رشته‌اطلاعات

پس از این مرحله باید تابع چگالی احتمال را از هیستوگرام (شکل ۳) ساخته شده به دست آورد. تابع چگالی احتمال به دست آمده از هیستوگرام ساخته شده باید شرط‌های (۱) و (۲) و (۳) را برآورده کند.

$$f_x(x) \geq 0 \quad (1)$$

$$f_x(x) \leq 1 \quad (2)$$

$$\sum_{-\infty}^{\infty} f_x(x) = 1 \quad (3)$$

در سه شرط (۱) و (۲) و (۳)، $f_x(x)$ مقدار تابع هیستوگرام ساخته شده است، که نشان‌دهنده تعداد تکرار اعداد بین ۰ تا ۲۵۵ است. با بررسی هیستوگرام، می‌توان نشان داد که شرط (۱) برآورده می‌شود، زیرا تعداد تکرار یک عدد همواره عددی بزرگ‌تر و یا مساوی با صفر خواهد بود (شکل ۳). اما تابع هیستوگرام به تنهایی شرط‌های (۲) و (۳) را برآورده نمی‌کند. برای برآورده کردن شرط‌های (۲) و (۳) باید تعداد تکرار برای هر عدد را بر مجموع کل تکرارهای به دست آمده برای آن عدد مشخص تقسیم کرد (معادله ۴). در این صورت شرط‌های

سال ۱۳۹۱ شماره ۲ پیاپی ۱۸

به وجود آورد. برای حل این موضوع می‌توان در زمان‌های مشخص، تابع چگالی احتمال را برای یک‌سری رشته تخمین زده و سپس تابع تخمین زده شده را به همراه رشته‌بیت اطلاعات به گیرنده ارسال کرد.

(جدول ۱): افزایش طول رشته‌اطلاعات پس از کد کردن با الگوریتم هافمن در صورت تخمین نادرست تابع چگالی احتمال با توجه به مثال شکل (۱)

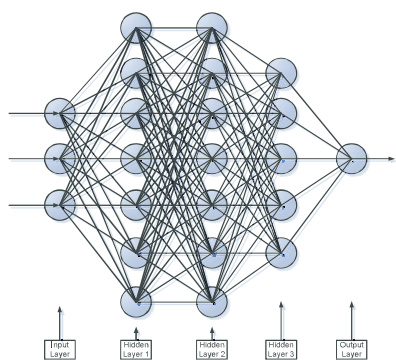
| طول رشته‌بیت خروجی پس از کد کردن با روش هافمن با تابع چگالی احتمال با میزان خطای بالا | طول رشته‌بیت ورودی |
|---|--------------------|
| ۲۴۲۶۶۶۶ (bit) | ۱۷۲۰۱۹۲ (bit) |

در گیرنده نیز ابتدا تابع چگالی احتمال ساخته می‌شود و با استفاده از آن، گیرنده شروع به کدگشایی اطلاعات می‌کند. اما مشکل بر سر نحوه تخمین تابع چگالی احتمال نمونه‌ها و نحوه ارسال آنهاست. در این مقاله یک روش نوین برای تخمین و ارسال تابع چگالی احتمال بر مبنای کد هافمن و استفاده از شبکه عصبی به منظور حافظه کردن رشته‌اطلاعات ارسالی برای افزایش میزان فشردگی ارائه شده است. همچنین سعی شده تا رشته‌بیت‌ها را حافظه‌دار کرد تا بتوان چندین بار الگوریتم هافمن را با توابع مختلف چگالی احتمال پیاده‌سازی کرد. در روش پیشنهادی به دلیل حافظه‌دار کردن این نوع کدگذاری به راحتی می‌توان رشته‌بیت‌ها را در گیرنده بازیابی کرد.

۳- نحوه به دست آوردن تابع چگالی احتمال برای یک رشته‌بیت با طول دلخواه

برای به دست آوردن تابع چگالی احتمال، نیاز به تعداد بیت زیادی است. بدین منظور در ابتدا طول مشخصی از بیت‌ها را از رشته‌اطلاعات جدا کرده یا اینکه سامانه منتظر می‌ماند تا بافر به اندازه طول مورد نظر پر شود. در مرحله بعد، پس از پر شدن بافر توسط بیت‌های رشته‌اطلاعات، هر هشت بیت از رشته مورد نظر به مبنای ۱۰ تبدیل شده و سپس به بافری دیگر انتقال داده می‌شوند (شکل ۲).

به نرون‌ها متصل می‌شوند، برقرار می‌شود. به این اتصال‌ها اصطلاحاً وزن‌های شبکه گفته می‌شود. در شکل ۵ نمونه‌ای از ساختاری یک شبکه عصبی با سه ورودی و سه لایه فرعی نشان داده شده است. لازم به ذکر است که علاوه بر وزن‌ها، هر نرون متأثر از کمیت مستقل دیگری با نام بایاس نیز است. نحوه تأثیرپذیری مقادیر وزن‌ها و بایاس‌ها در نرون‌های هر لایه، به نرون‌های لایه بعدی توسط تابعی تحت عنوان تابع انتقال، مشخص می‌شود. تابع انتقال در هر لایه می‌تواند خطی و یا غیر خطی باشد که توسط کاربر تعیین می‌شود. علاوه بر این، وزن‌ها و بایاس‌های مربوط به هر نرون، عامل مهم و حائز اهمیت در شبکه هستند که مقادیر آنها در نتایج خروجی شبکه عصبی تأثیر گذار است.



(شکل ۵): نمایش ساختار یک شبکه عصبی چند لایه

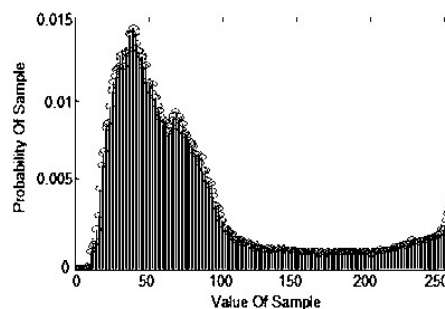
لذا در روش‌های متعددی که جهت بهبود دقت شبکه‌های عصبی به کار رفته است، سعی بر انتخاب مقادیر مناسب این وزن‌ها و بایاس‌ها، به گونه‌ای است که کم‌ترین خطای شبکه را در برداشته باشد. به این عمل به اصطلاح آموزش شبکه گفته می‌شود.

۴-۱- نحوه استفاده از شبکه عصبی برای توصیف تابع چگالی احتمال

در این قسمت هدف افزایش کردن تابعی بر منحنی چگالی احتمال (شکل ۴) با استفاده از شبکه عصبی است. (Cobourn, et al., 2009). برای این کار ابتدا یک شبکه عصبی با یک لایه پنهان آموزش ندیده، ایجاد می‌شود. سپس تمام وزن‌ها و بایاس‌های شبکه به عنوان متغیرهای بهینه‌سازی معرفی می‌شوند. ورودی شبکه عصبی مقادیر محور افقی تابع چگالی احتمال (اعداد صحیح بین ۰ تا ۲۵۵) و خروجی شبکه عصبی مقادیر محور عمودی تابع چگالی احتمال (احتمال رخداد نمونه‌ها) در نظر گرفته می‌شوند. تابع

(۲) و (۳) نیز برآورده می‌شوند. با این کار می‌توان منحنی تابع چگالی احتمال (شکل ۴) را با استفاده هیستوگرام (شکل ۳) به دست آورد.

$$f_x(x_i) = \frac{f_x(x_i)}{\sum_{j=0}^{255} f_x(x_j)} \quad (4)$$



(شکل ۴): منحنی تابع چگالی احتمال

تا این مرحله تابع چگالی احتمال مربوط به تعداد تکرار اعداد ساخته شده در یک رشته به دست آورده می‌شود. اما با توجه به اینکه تابع چگالی احتمال به دست آمده برای هر رشته، شکلی متفاوت از رشته‌های دیگر دارد، برای کد کردن این رشته‌ها با الگوریتم هافمن باید تابع چگالی احتمال نیز همراه با رشته اطلاعات مربوطه ارسال شود. یکی از روش‌هایی که می‌توان برای ارسال تابع چگالی احتمال استفاده کرد، این است که بر روی آن نموداری افزای کرده و اطلاعات آن نمودار را به همراه رشته اطلاعات ارسال کرد. اکنون این سؤال پیش می‌آید که با توجه به تغییرات سریع و غیر خطی تابع چگالی احتمال (شکل ۴) چه نموداری می‌توان بر روی آن افزای کرد به وجهی که کمترین خطا را با تابع چگالی احتمال مورد نظر داشته باشد. برای این کار می‌توان از شبکه عصبی مصنوعی کمک گرفت که در بخش چهار به صورت خلاصه به توضیح آن می‌پردازیم.

۴- شبکه عصبی مصنوعی

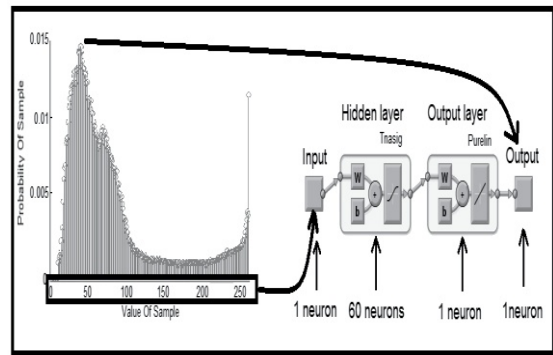
شبکه عصبی مصنوعی از یک سری لایه‌ها و گره‌ها تشکیل می‌شود. هر شبکه به‌الزام دارای لایه ورودی و خروجی است (Vemuri, 1988, Gardner, et al., 1998). در هر شبکه می‌توان به دلخواه تعدادی لایه فرعی اضافه کرد. هر لایه شامل تعداد دلخواهی گره است که تحت عنوان نرون‌ها شناخته می‌شوند. ارتباط بین هر لایه توسط اتصال‌هایی که

وزن‌ها و بایاس‌های به‌روزشده، به‌عنوان وزن‌ها و بایاس‌های شبکه، تثبیت می‌شوند. ولی اگر خطا بیشتر از حد مشخص شده باشد، آموزش دوباره تکرار می‌شود. این کار از Over Learning یا به‌اصطلاح آموزش بیش از حد جلوگیری می‌کند. اما در این مقاله برخلاف روش‌های معمول، سعی بر این است که شبکه به‌طور کامل بر مقادیر احتمال رخداد نمونه‌ها منطبق شود و به‌اصطلاح Overlearning رخ دهد. برای اینکه بتوان به این هدف رسید، تمامی مقادیر احتمال رخداد نمونه‌ها به‌عنوان نقاط آموزشی به شبکه عصبی وارد می‌شوند و در نتیجه پس از چندین بار به‌روزشدن مقادیر وزن‌ها و بایاس‌ها، مقادیر تابع به‌دست آمده با استفاده از شبکه عصبی تا حد زیادی به منحنی چگالی احتمال نزدیک می‌شود.

۲-۴- پیاده سازی شبکه عصبی برای تابع چگالی احتمال

به‌دلیل متغیر با زمان بودن منحنی‌های چگالی احتمال و همچنین به‌دلیل تغییرات زیاد در این منحنی‌ها نمی‌توان توابعی را با استفاده از روش‌های کلاسیک برای افزایش کردن بر روی این منحنی‌ها به‌دست آورد. بنابراین در این مقاله از شبکه عصبی برای افزایش کردن تابعی بر روی منحنی چگالی احتمال استفاده شده است. این امر باعث می‌شود که مقادیر تابع محاسبه شده ذخیره شود و از آن برای حافظه‌دار کردن توابع چگالی احتمال استفاده شود. برای مثال اگر شبکه عصبی ساخته شده با شصت نرون در لایه پنهان را که دارای ۱۲۰ وزن و ۶۱ بایاس باشد^۱ برای رشته‌اطلاعاتی که دارای منحنی چگالی احتمالی شبیه به شکل (۴) است آموزش دهیم، خروجی در بهترین حالت با فرض (خطای میانگین مربعی کمتر از 10^{-10}) در شکل (۷) و شکل (۸) نشان داده شده است. در شکل (۷) تابع هدف و مقادیر خروجی به‌دست آمده با استفاده از شبکه عصبی با یکدیگر رسم شده‌اند. همان‌طور که در این شکل دیده می‌شود مقادیر خروجی با خطای بسیار کمی بر روی مقادیر تابع هدف منطبق شده‌اند.

هدف در شبکه عصبی به‌کار گرفته شده، بر اساس میانگین مربعات خطا بین احتمال رخداد نمونه‌های تابع چگالی احتمال و مقادیر محاسبه شده با توجه به وزن‌ها و بایاس‌های شبکه عصبی معرفی می‌شود. بنابراین در هر مرحله از بهینه‌سازی، مقادیر وزن‌ها و بایاس‌های شبکه، به‌گونه‌ای به روز می‌شوند که خطای شبکه کاهش یابد. در این مقاله برای افزایش کردن تابعی بر نمودار چگالی احتمال از شبکه عصبی پرسپترون با یک لایه پنهان که شصت نرون دارد، استفاده شده است. تابع انتقال نرون‌های لایه پنهان tansig و تابع انتقال لایه خروجی purelin است. همچنین لایه ورودی و خروجی، تک‌نرونی هستند و تمامی نرون‌های هر سه لایه دارای بایاس هستند (شکل ۶).



(شکل ۶): ساختار شبکه استفاده شده در این مقاله با فرض ۶۰ نرون در لایه پنهان شبکه

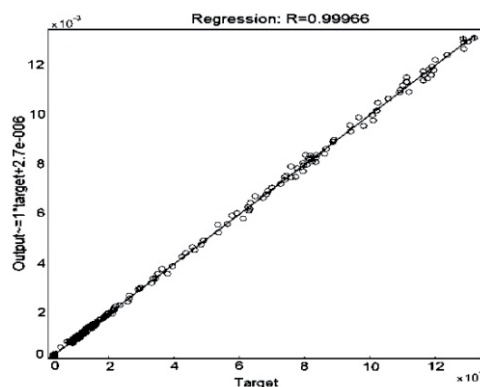
همان‌طور که در شکل (۶) نشان داده شده است ورودی شبکه عصبی مقادیر محور افقی و تابع هدف شبکه عصبی مقادیر عمودی متناظر با مقادیر افقی است. وزن‌های شبکه تا زمانی که خطا از 10^{-10} کمتر شود، به‌روز می‌شوند. باید به این نکته اشاره کرد که نوع تابع انتقال با روش Leave-One-Out پس از امتحان بر روی چندین تابع چگالی احتمال انتخاب شده است. در این روش هدف ما در آموزش دادن شبکه عصبی برخلاف آموزش‌های معمول است. به این معنی که در آموزش‌های متداول، درصدی از اطلاعات به‌عنوان داده‌های آزمون کنار گذاشته می‌شوند و در انتهای هر بار آموزش و به‌روزشدن وزن‌ها و بایاس‌ها، داده‌های آزمون به‌عنوان ورودی به شبکه عصبی وارد می‌شوند و خروجی‌های به‌دست آمده با مقادیر صحیح مقایسه می‌شوند. اگر خطا از حد معینی کمتر باشد،

^۱ تعداد ۶۰ بایاس برای لایه پنهان و یک بایاس برای لایه خروجی

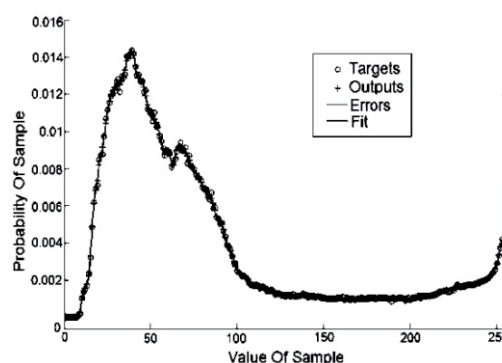
می‌توان تابع چگالی احتمال را با وزن‌ها و بایاس‌های به‌دست آمده از آموزش شبکه عصبی توصیف و همچنین رفتار تابع چگالی احتمال را با استفاده از مجموعه وزن‌ها و بایاس‌های شبکه عصبی می‌توان ذخیره‌سازی کرد. برای ذخیره هر کدام از وزن‌ها و بایاس‌ها دو بایت نیاز است. در ادامه مزیت حافظه‌دار کردن رشته‌اطلاعات بیان می‌شود.

۵-۱- مزیت حافظه‌دار کردن رشته‌اطلاعات

با استفاده از حافظه‌دار کردن رشته‌اطلاعات می‌توان الگوریتم هافمن را چندین بار بر روی رشته‌اطلاعات پیاده‌سازی کرد. بدین صورت که قبل از هر بار ارسال اطلاعات، تابع چگالی احتمال با استفاده از شبکه عصبی تقریب زده می‌شود و سپس الگوریتم هافمن با استفاده از مقادیر خروجی شبکه عصبی که برای تابع چگالی احتمال به‌دست آمده است، اجرا می‌شود. وزن‌ها و بایاس‌های به‌روزشده به‌وسیله شبکه عصبی، در انتها به رشته‌اطلاعات اضافه می‌شوند. با تکرار این روش می‌توان حجم رشته‌اطلاعات ارسالی را بسیار کم کرد؛ اما تعداد دفعات تکرار این الگوریتم برای فشرده‌سازی رشته‌اطلاعات رابطه مستقیم با طول رشته‌اطلاعات دارد. این بدان معناست که اگر بخواهیم چندبار این الگوریتم را اجرا کنیم، با توجه به اینکه در هر بار اجرا کردن این الگوریتم، باید تابع چگالی احتمال توسط شبکه عصبی تخمین زده شده و تعداد بیتی که بیان‌گر پارامترهای شبکه عصبی به‌کار گرفته شده است با استفاده از معادله (۶) (که در بخش ۶-۲ آورده شده است) محاسبه شده و به انتهای رشته‌اطلاعات اضافه می‌شود، که این امر موجب افزایش طول رشته‌اطلاعات می‌شود. در اینصورت اگر طول رشته‌اطلاعات اولیه کوتاه باشد، نمی‌توان این الگوریتم را چندین بار برای آن رشته به‌کار برد؛ زیرا تعداد بیت استفاده شده برای معرفی تابع چگالی احتمال به مراتب بیشتر از طول رشته‌اطلاعات است. بنابراین تعداد تکرار این الگوریتم محدود است و رابطه مستقیم با طول رشته دارد یعنی برای رشته‌هایی با طول زیاد تعداد تکرار می‌تواند بیشتر از تعداد تکرار برای رشته‌هایی با طول کوتاه‌تر باشد. در ادامه الگوریتم فشرده‌سازی گام به گام توضیح داده می‌شود.



(شکل ۷): رسم همزمان مقادیر تابع هدف همراه با مقادیر خروجی به دست آمده با شبکه عصبی



(شکل ۸): رسم میزان پراکندگی مقادیر هدف از مقادیر خروجی

شکل (۸) مقدار رگرسیون^۱ را نشان می‌دهد که دلیلی بر صحت میزان خطای ناچیز داده‌های خروجی است. البته باید به این نکته توجه کرد که در مواقعی که نمودار با شیب تند نزول می‌کند و مقادیر به سمت صفر میل می‌کنند، ممکن است که تابع تقریب زده مقادیری کمتر از صفر را برای این نقاط تقریب بزنند؛ که شرط (۱) را نقض می‌کند.

در این مواقع مقادیر منفی با کم‌ترین مقدار مثبت تابع تخمین زده شده جایگزین می‌شوند؛ زیرا نقاطی که با مقادیر منفی تخمین زده شده‌اند، دارای مقادیر ناچیزی دارند و لذا می‌توان آنها را با کم‌ترین مقدار مثبت تابع تخمین زده شده جایگزین کرد.

۵- مفهوم حافظه‌دار کردن رشته‌اطلاعات

با توجه به مطالب گفته‌شده در بخش‌های ۴-۱ و ۴-۲ می‌توان گفت که تابع چگالی احتمال را با استفاده از شبکه عصبی مصنوعی می‌توان توصیف کرد. این بدان معناست که

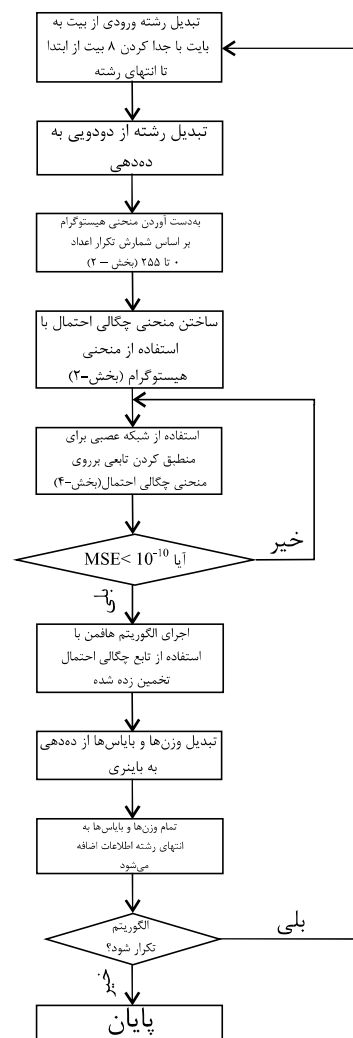
^۱ میزان پراکندگی مقادیر هدف از مقادیر خروجی

۶- توضیح الگوریتم فشرده‌سازی

۱-۶ نحوه کد کردن

شکل (۹) نمودار جریان‌ی مربوط به الگوریتم فشرده‌سازی را نشان می‌دهد. مراحل انجام این الگوریتم به شرح ذیل است:

- مرحله ۱: ابتدا رشته اطلاعات از مبنای دودویی به مبنای دهدهی تبدیل می‌شود. (بخش ۳)
- مرحله ۲: منحنی هیستوگرام بر اساس تعداد تکرار اعداد ۰ تا ۲۵۵ در رشته اطلاعات ساخته شده در مرحله ۱ به دست آورده می‌شود. (بخش ۳)
- مرحله ۳: به دست آوردن منحنی چگالی احتمال با



(شکل ۹): رسم نمودار جریان‌ی الگوریتم کد کردن

- استفاده از منحنی هیستوگرام، با برآورده کردن شرط‌های (۱) و (۲) و (۳). (بخش ۳)
- مرحله ۴: استفاده از شبکه عصبی برای افزایش کردن تابعی بر روی منحنی چگالی احتمال. (بخش ۴)
- مرحله ۵: اجرای الگوریتم هافمن با استفاده از تابع چگالی احتمال تخمین زده شده در مرحله ۴.
- مرحله ۶: تبدیل وزن‌ها و بایاس‌ها از مبنای دهدهی به مبنای دودویی و اضافه کردن آنها به انتهای رشته اطلاعات فشرده شده جهت ارسال.
- مرحله ۷: در صورت نیاز به تکرار الگوریتم جهت فشرده‌سازی بیشتر، مراحل ۱ تا ۶ دوباره تکرار شده و در انتها تعداد دفعات تکرار الگوریتم به همراه رشته اطلاعات ارسال می‌شود.

۶-۲ نحوه کدگشایی کردن و بازیابی اطلاعات

شکل (۱۰) نمودار جریان‌ی الگوریتم بازیابی اطلاعات را نشان می‌دهد. مراحل بازیابی اطلاعات به شرح ذیل است:

- مرحله ۱: جداسازی H بیت به دست آمده از معادله (۶) از انتهای رشته اطلاعات.

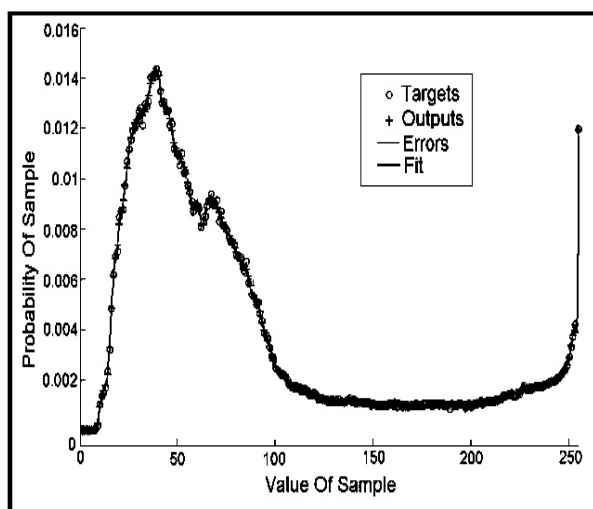
$$H = \left(\frac{2 * \text{Byte}}{\text{نرون}} * \frac{8 \text{ bit}}{\text{Byte}} \right) * (1 + \text{پنهانی}) \quad (6)$$

- مرحله ۲: H بیت جدا شده را از مبنای دودویی به مبنای دهدهی تبدیل کرده و در نتیجه وزن‌ها و بایاس‌های آخرین تابع چگالی احتمال را به دست آوریم.

- مرحله ۳: با استفاده از وزن‌ها و بایاس‌های تولید شده از مرحله ۲ و با استفاده از شبکه عصبی تابع چگالی احتمال دوباره تولید می‌شود.

- مرحله ۴: با توجه به تابع چگالی احتمال به دست آمده رشته اطلاعات با استفاده از الگوریتم هافمن کدگشایی می‌شود.

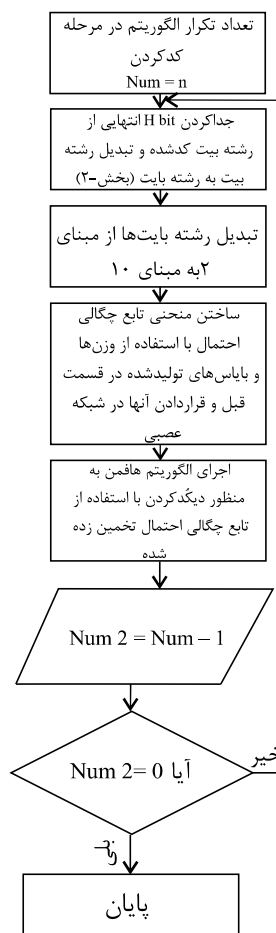
- مرحله ۵: این الگوریتم به تعداد دفعاتی که در فرستنده اجرا شده در گیرنده نیز باید اجرا شود. برای اجرای دوباره الگوریتم کدگشایی، باید به مرحله ۱ بازگردیم. زمانی که تعداد دفعات اجرای الگوریتم کدگشایی برابر با تعداد دفعات اجرای الگوریتم در زمان کدگشایی شود، الگوریتم کدگشایی خاتمه می‌یابد.



(شکل ۱۱): تابع چگالی احتمال تخمین زده شده در اولین اجرا الگوریتم

۸- ارزیابی روش پیشنهادی

قبل از ارزیابی الگوریتم پیشنهادی به صورت اجمالی در مورد ISCAS benchmark circuits توضیحی ارائه داده و سپس به ارزیابی روش پیشنهادی می پردازیم. ISCAS benchmark circuits شامل مداراتی هستند که با استفاده از درگاه منطقی شروع به تولید رشته بیت هایی با طول مشخص می کنند. هر یک از این مدارات ساختاری متفاوت داشته و رشته بیت های تولیدی آنها نیز دارای طول متفاوت از دیگری است. این مدارات به صورت نرم افزار برای FPGA ها نیز نوشته شده است. در این مقاله فایل چند سری از مداراتی که در مرجع (Chandra, et al., 2003) مورد استفاده قرار گرفته اند، اجرا شده و رشته بیت خروجی آنها توسط الگوریتم پیشنهادی فشرده سازی شده و با روش های FDR و Golomb از لحاظ درصد فشرده سازی مقایسه شده است. به منظور ارزیابی روش پیشنهادی دو آزمون با استفاده از ISCAS benchmark circuits ترتیب داده شده است. در آزمون اول از یک رشته بیت ساخته شده با استفاده از ATPG (Hamzaoglu, et al., 1998) و در آزمون دوم از رشته بیت ساخته شده برای ISCAS benchmark circuit از HITEC (Univ. of Illinois IGATE website) استفاده شده است. در آزمون اول درصد فشرده سازی بین روش پیشنهادی با روش های FDR (Chandra, et al., 2003) و Golomb مقایسه شده است و در آزمون دوم درصد فشرده سازی تنها بین روش پیشنهادی با روش FDR مقایسه شده است؛ زیرا روش



(شکل ۱۰): رسم نمودار جریان الگوریتم کدگشایی

۷- استفاده از الگوریتم ارائه شده برای فشرده سازی اطلاعات

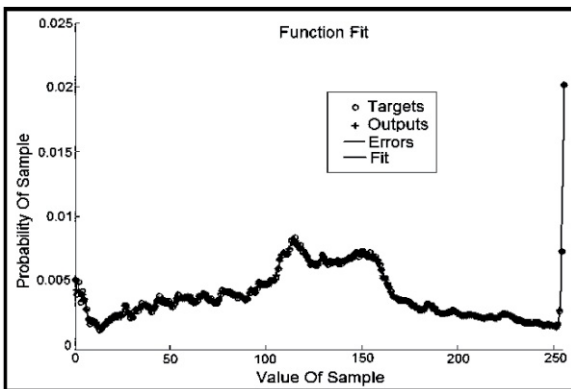
به عنوان مثال، شکل های (۲) و (۳) و (۴) با انجام مراحل ۱، ۲، ۳ و ۴ از بخش ۶-۱ به دست آمده اند. در مرحله بعد الگوریتم هافمن با توجه به تابع چگالی احتمال به دست آمده از شبکه عصبی اجرا می شود و رشته اطلاعات را فشرده سازی می کند. در این مثال، الگوریتم روش پیشنهادی چهار مرتبه اجرا شده است و نتایج آن در جدول (۲) و شکل های ۱۴ و ۱۳، ۱۲، ۱۱ آورده شده است. همان طور که در منحنی های ۱۴ و ۱۲، ۱۳ دیده می شود مقادیر منفی در تخمین تابع نیز وجود دارند که همان طور که در بخش ۴-۲ گفته شد این مقادیر با کوچکترین مقدار مثبت تابع چگالی احتمال جایگزین شده و با تابع چگالی احتمال ساخته شده، الگوریتم هافمن اجرا می شود.

مرتبه اجرای الگوریتم در جدول (۳) گزارش شده است، که بهترین آنها با درصد فشرده‌سازی روش‌های FDR و Golomb مقایسه شده‌اند.

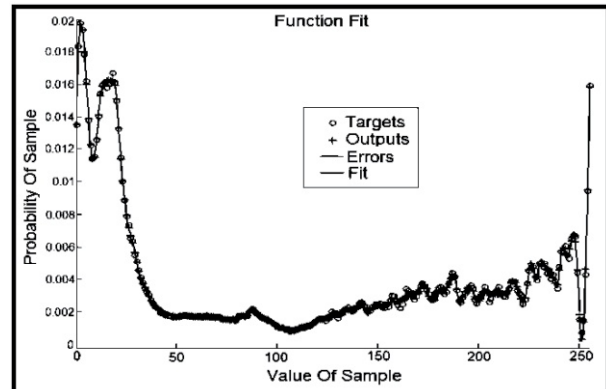
FDR دارای درصد فشرده‌سازی بیشتری نسبت به روش Golomb است و رشته‌اطلاعات را بیشتر فشرده می‌کند. پس از شش مرتبه اجرای الگوریتم روش پیشنهادی، درصد فشرده‌سازی رشته‌اطلاعات به‌ازای ۶ و ۴ و ۳ و ۲ و ۱ $M =$

(جدول ۲): نمایش میزان فشردگی برای چهار حالت که در آنها دفعات تکرار الگوریتم متفاوت است

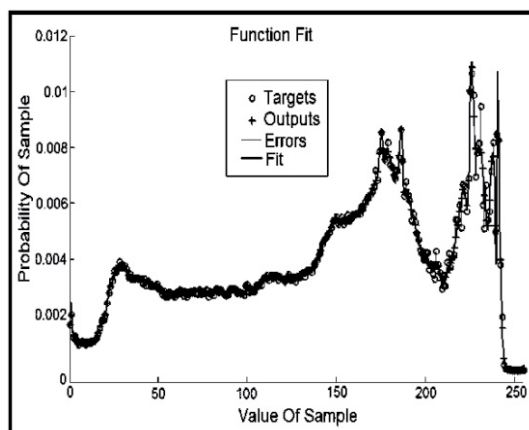
| اندازه رشته ورودی (bit) ۱۶۹۹۲۰۰ | دفعات تکرار الگوریتم $M=$ $H= 2896 \text{ bit}$ و $60 =$ تعداد نرون لایه پنهان | | | | |
|------------------------------------|---|---------|--------|--------|--------|
| | $M=1$ | $M=2$ | $M=3$ | $M=4$ | $M=8$ |
| اندازه رشته ساخته شده | ۱۶۵۱۵۳۱ | ۲۴۷۴۵۲۳ | ۱۴۲۲۵۶ | ۷۰۵۹۵ | ۳۸۹۱ |
| درصد فشرده سازی | %۲/۸ | %۷۲/۰۷ | %۹۱/۶۲ | %۹۵/۸۴ | %۹۸/۵۳ |



(شکل ۱۴): تابع چگالی احتمال تخمین زده شده در چهارمین اجرای الگوریتم



(شکل ۱۲): تابع چگالی احتمال تخمین زده شده در دومین اجرای الگوریتم



(شکل ۱۳): تابع چگالی احتمال تخمین زده شده در سومین اجرای الگوریتم

۸-۱-آزمون اول

FDR مربوط به رشته S38584 با مقدار $\frac{33}{1}$ است و برای Golomb مربوط به رشته S38584 با مقدار $\frac{38}{16}$ و کمترین اختلاف مقدار فشردگی بین روش پیشنهادی با روش FDR مربوط به رشته S13207 با مقدار $\frac{9}{5}$ است و برای Golomb مربوط به رشته S13207 با مقدار $\frac{12}{84}$ است.

مقایسه درصد فشردگی برای رشته‌بیت ساخته‌شده با استفاده از ATPG بین روش پیشنهادی با روش‌های FDR و Golomb همان‌طور که در جدول (۳) دیده می‌شود، بیش‌ترین اختلاف مقدار فشردگی بین روش پیشنهادی با روش

(جدول ۳): مقایسه درصد فشردگی سازی

| Circuit | درصد فشردگی سازی Golomb | درصد فشردگی سازی FDR | درصد فشردگی سازی با پیاده سازی الگوریتم معرفی شده در این مقاله با M بار تکرار الگوریتم تعداد نرون لایه پنهان = 60 و H=2896 bit | | | | | بهترین درصد فشردگی سازی | اندازه رشته‌اطلاعات (bit) | اندازه رشته کد شده (bit) |
|---------|----------------------------|-------------------------|--|-------|-------|-------|-------|-------------------------|---------------------------|--------------------------|
| | | | M=1 | M=2 | M=3 | M=4 | M=6 | | | |
| S5378 | 53/73 | 61/32 | 56/14 | 72/59 | 80/42 | 87/42 | 82/39 | 82/39 | 23754 | 4174 |
| S9234 | 59/85 | 60/63 | 61/48 | 81/88 | 87/35 | 86/67 | 88/39 | 88/39 | 39273 | 4559 |
| S13207 | 84/33 | 87/67 | 68/17 | 89/72 | 83/63 | 96/25 | 97/17 | 97/17 | 165200 | 4675 |
| S15850 | 66/55 | 71/95 | 35/34 | 67/75 | 89/13 | 93/02 | 94/54 | 94/54 | 76986 | 4203 |
| S38417 | 58/08 | 65/35 | 53/79 | 72/67 | 94/14 | 96/22 | 94/97 | 96/22 | 164736 | 6227 |
| S38584 | 59/61 | 64/67 | 68/86 | 89/54 | 85/53 | 95/24 | 97/77 | 97/77 | 199104 | 4440 |

میزان تکرار برای رسیدن به بهترین درصد فشردگی سازی در رشته‌هایی با طول زیاد نسبت به رشته‌هایی با طول پایین می‌تواند بیشتر باشد. برای روشن شدن مطلب و وضوح بیشتر شکل (۱۵)، نمودار میله‌ای مربوط به جدول (۳) را جهت مقایسه بهتر نشان می‌دهد.

۸-۲- آزمون دوم

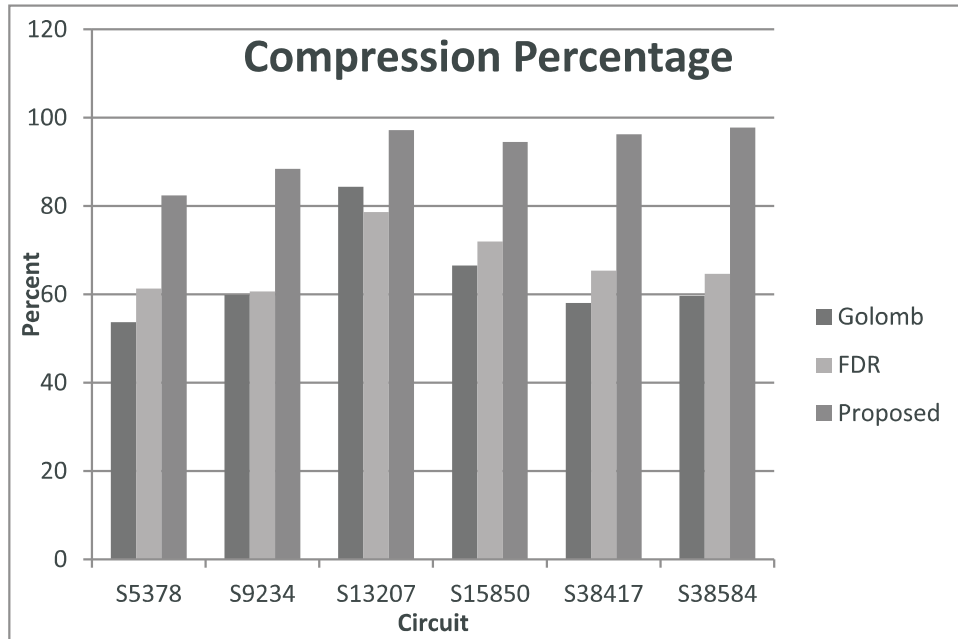
مقایسه درصد فشردگی سازی رشته‌بیت ساخته‌شده برای ISCAS benchmark circuits با استفاده از HITEC بین روش پیشنهادی با روش FDR:

در این آزمایش با توجه به اینکه طول رشته‌اطلاعات S953 بسیار کم است، روش پیشنهادی قادر به فشردگی سازی با درصد بالا را نخواهد داشت که علت آن در بخش ۸-۱ توضیح داده شد. در بهترین حالت درصد فشردگی سازی برای

روش پیشنهادی برای رشته‌هایی با طول کم (مانند S953) قادر به فشردگی سازی در حد بالایی نیست، زیرا افزایش تکرارهای الگوریتم پیشنهادی، به علت اضافه شدن اطلاعات مربوط به تابع چگالی احتمال در انتهای رشته کدشده، باعث کم شدن درصد فشردگی سازی می‌شود. اما برای رشته‌هایی با طول زیاد (مانند S35932)، درصد فشردگی سازی با افزایش تکرار الگوریتم پیشنهادی افزایش می‌یابد. دلیل این امر آن است که داده‌های مربوط به شبکه عصبی، برای رشته‌های با طول زیاد نسبت به طول رشته ارسالی ناچیز است. تعداد تکرارهای الگوریتم پیشنهادی باید محدود باشد، زیرا پس از چندین تکرار درصد فشردگی سازی به دلیل افزایش داده‌های مربوط به ذخیره سازی شبکه عصبی، کاهش پیدا می‌کند. اما

به فشرده‌سازی رشته‌بیت‌های اعمال شده است، بنابراین اگر پس از رسیدن به بهترین درصد فشرده‌سازی، دوباره الگوریتم پیشنهادی اجرا شود و یا لایه‌های پنهان شبکه عصبی به کار رفته در الگوریتم پیشنهادی افزایش یابد، به دلیل افزایش اطلاعات مربوط به وزن‌های شبکه عصبی، درصد فشرده‌سازی رشته‌بیت اطلاعات ارسالی کاهش می‌یابد.

رشته S953 معادل با ۶۸/۴۷٪ است، که برای شبکه عصبی با سه نرون در لایه پنهان و چهار تکرار به دست آمده است. ولی برای رشته S13207 با تغییر تعداد نرون لایه پنهان و تعداد تکرار الگوریتم نتایج جدول (۵) به دست می‌آید. همان‌طور که در جدول (۵) مشاهده می‌شود به‌ازای مقادیر $N=4$ و $M=14$ درصد فشرده‌سازی کاهش یافته است. دلیل این امر آن است که الگوریتم پیشنهادی تا میزان خاصی قادر



(شکل ۱۵): نمایش میله ای مقایسه درصد فشرده سازی برای جدول (۳)

Golomb صورت پذیرفته که بهترین درصد فشرده‌سازی در $\text{scan vector} = 30$ به دست آمده است. مقادیر درصد فشرده‌سازی این آزمایش در جدول (۶) آورده شده است. با استفاده از الگوریتم پیشنهادی ۵۴٪ درصد فشرده‌سازی را در مقایسه با روش FDR و ۱/۴۲٪ درصد فشرده‌سازی را در مقایسه با روش Golomb افزایش داده‌ایم. لازم به ذکر است همان‌گونه که در جدول (۴) مشاهده می‌شود، درصد فشرده‌سازی به‌طور عمومی بالای ۹۹٪ است و لذا افزایش ۵۴٪ فشرده‌سازی نیز قابل قبول است. درصد فشرده‌سازی با استفاده از معادله (۷) حساب می‌شود.

$$(7) \quad \text{درصد فشرده‌سازی} = \frac{\text{طول رشته کد شده} - \text{رشته طول ورودی}}{\text{رشته ورودی طول}} * 100$$

زیرا اجرای مجدد شبکه عصبی و یا افزایش نرون‌ها فقط بیت‌های ارسالی به‌منظور بازسازی شبکه عصبی را بیشتر می‌کند ولی میزان فشرده‌سازی رشته‌بیت ورودی افزایش نمی‌یابد. اما نتایج روش پیشنهادی برای سایر رشته‌ها به مراتب خیلی بهتر از روش FDR بوده است. در بهترین حالت درصد فشرده‌سازی به‌میزان ۱۶/۰۵٪ برای رشته S5378 بهبود یافته است. برای رشته‌های دیگر نیز درصد فشرده‌سازی در جدول (۴) نشان داده شده است. شکل (۱۶) نمودار میله‌ای مربوط به جدول (۴) را جهت مقایسه بهتر و وضوح بیشتر نشان می‌دهد.

۳-۸ - مقایسه درصد فشرده‌سازی با بهترین مقدار به دست آمده از $\text{scan vector} = 30$

در (Chandra, et al., 2003)، ۳۲ مرحله فشرده‌سازی روی رشته‌بیتی با طول ۳۶۲۹۲۲ بیت با روش‌های FDR و

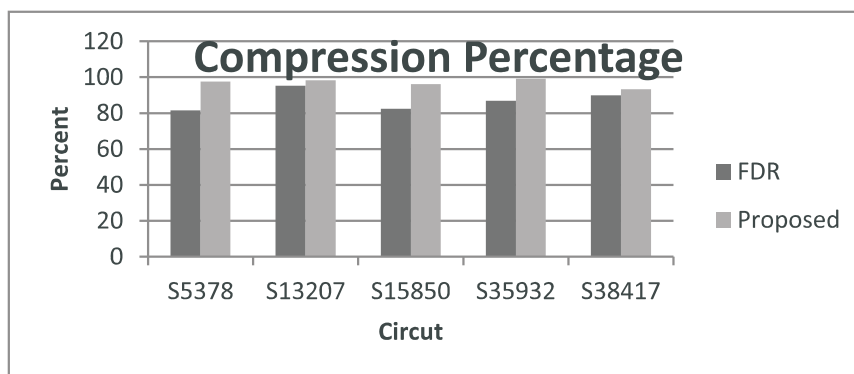
۹- نتیجه‌گیری

در این مقاله روشی جدید برای کدکردن رشته‌اطلاعات معرفی شد. این نوع کدگذاری بر روی رشته‌بیت‌های ساخته شده با استفاده از ISCAS benchmark circuits اعمال شد. نتایج به‌دست آمده از میزان فشردگی با دو کد FDR و Golomb مقایسه شد، که بیش‌ترین اختلاف مقدار فشردسازی بین روش پیشنهادی با روش FDR معادل با ۳۳/۱٪ و برای روش Golomb معادل با ۳۸/۱۶٪ به دست آمد و کمترین اختلاف مقدار فشردسازی بین روش پیشنهادی با روش FDR معادل با ۹/۵٪ و برای روش Golomb معادل با ۱۲/۸۴٪ به‌دست آمد. البته این نوع

کدگذاری برای رشته‌هایی با طول پایین، مناسب نیست. به‌دلیل اینکه پس از هر بار اجرای الگوریتم پیشنهادی تعدادی بیت که برای فرستادن تابع چگالی احتمال مورد نیاز است به انتهای رشته کدشده اضافه می‌شوند. همچنین برای رشته‌اطلاعات مورد نظر با طول ۳۶۲۹۲۲ بیت، بهترین درصد فشردسازی به‌وسیله روش پیشنهادی که با کم‌کردن نرون‌های لایه پنهان شبکه عصبی به منظور کاهش تعداد بیت‌های اضافه شده به انتهای رشته کدشده و افزایش تعداد اجرای الگوریتم نسبت به بهترین درصد فشردسازی با دو روش FDR و Golomb به‌دست آمده است.

(جدول ۴): مقایسه درصد فشردسازی FDR با الگوریتم این مقاله با استفاده از HITEC

| Circuit | بهترین درصد فشردسازی FDR | درصد فشردسازی با پیاده‌سازی الگوریتم معرفی شده در این مقاله با M بار تکرار الگوریتم تعداد نرون لایه پنهان $= 60$ و $H=2896$ bit | | | | | بهترین درصد فشردسازی | اندازه رشته اطلاعات (bit) | اندازه رشته کدشده (bit) |
|---------|--------------------------|---|-------|-------|-------|-------|----------------------|---------------------------|-------------------------|
| | | M=۱ | M=۲ | M=۳ | M=۴ | M=۶ | | | |
| S953 | ۷۵/۹۴ | -- | -- | -- | -- | -- | -- | ۱۱۶۸ | -- |
| S5378 | ۸۱/۴۵ | ۵۷/۱۹ | ۸۴/۶۷ | ۹۰/۷۷ | ۹۵/۵۳ | ۹۷/۵۰ | ۹۷/۵۰ | ۱۶۹۹۹۵ | ۴۲۴۹ |
| S13207 | ۹۵/۱۷ | ۶۵/۴۲ | ۸۲/۹۴ | ۸۶/۵۲ | ۸۶/۱۲ | ۹۸/۲۳ | ۹۸/۲۳ | ۴۲۲۸۴ | ۴۵۵۴ |
| S15850 | ۸۲/۴۱ | ۶۷/۹۱ | ۸۹/۰۴ | ۹۰/۷۷ | ۹۴/۱۲ | ۹۶/۲۰ | ۹۶/۲۰ | ۱۴۷۰۷۰ | ۵۵۸۸ |
| S35932 | ۸۶/۸۴ | ۶۸/۴۱ | ۷۰/۵۰ | ۹۳/۷۷ | ۹۶/۱۶ | ۹۸/۷۶ | ۹۸/۷۶ | ۴۳۰۳۵۸ | ۵۳۳۶ |
| S38417 | ۸۹/۸۶ | ۶۹/۷۷ | ۸۰/۴۳ | ۹۰/۱۱ | ۹۳/۳۸ | ۹۱/۰۴ | ۹۳/۳۸ | ۲۲۶۲۴ | ۱۴۹۷ |



(شکل ۱۶): نمودار میله ای مقایسه درصد فشردسازی برای جدول (۴)

(جدول ۵): محاسبه درصد فشرده‌سازی برای غلبه بر کد FDR با تغییر تعداد نرون‌های لایه پنهان و تعداد تکرار الگوریتم

| Circuit | بهترین درصد فشرده سازی FDR | درصد فشرده‌سازی با پیاده‌سازی الگوریتم معرفی شده در این مقاله با M بار تکرار الگوریتم و دارای N نرون در لایه پنهان شبکه | | | | بهترین درصد فشرده سازی | اندازه رشته اطلاعات (bit) | اندازه رشته کد شده (bit) |
|---------|----------------------------|---|---------------|---------------|---------------|------------------------|---------------------------|--------------------------|
| | | M = ۱۰ N=۳ | M = ۱۲ N=۳ | M = ۱۴ N=۳ | M = ۱۴ N=۴ | | | |
| S13207 | ۹۵/۱۷ | ۹۵/۱۸ | ۹۸/۲۷ | ۹۹/۱۱ | ۸۴/۹۷ | ۹۹/۱۱ | ۴۲۲۸۴ | ۳۷۶ |

(جدول ۶): مقایسه درصد فشرده‌سازی روش پیشنهادی با روش فشرده‌سازی در (Chakrabarty, et al., 2001)

| اندازه رشته اطلاعات (bit) | بهترین درصد فشرده‌سازی Golomb | بهترین درصد فشرده‌سازی FDR | بهترین درصد فشرده‌سازی الگوریتم معرفی شده در این مقاله با M بار تکرار الگوریتم و دارای N نرون در لایه پنهان شبکه M=۳ و N=۱۵ | اندازه رشته کد شده (bit) |
|---------------------------|-------------------------------|----------------------------|--|--------------------------|
| ۳۶۲۹۲۲ | ۹۸/۴۶ | ۹۹/۳۴ | ۹۹/۸۸ | ۴۳۵ |

Gardner, M.W., Dorling, S.R., 1998. Artificial Neural Networks (The Multilayer Perceptron), A Review of Applications in Atmospheric Sciences, Atmos, Environ.

Hamzaoglu, I., Patel, J.H., 1998. Test set compaction algorithms for combinational circuits, proc. int'l conf. computer-aided design, pp. 283-289.

Huffman, D.A., 1952. A method for the construction of minimum redundancy codes, Proc. IRE, Vol. 40, pp. 1098-1101.

Iyengar, V., Chakrabarty, K., Murray, B.T., 1999. Deterministic built-in self-testing of sequential circuits using pre computed test sets, J. Electronic Testing Theory and Applications (JETTA), Vol. 15, pp. 97-114.

Jas, A., Ghosh-Dastidar, J., Toubia, N.A., 1999. Scan vector compression decompression using statistical coding, Proc. IEEE VLSI Test Symp, pp. 114-120.

Proakis, J.G., 1995. Digital Communications, Mcgraw Hill, Hardcover.

۱۰ - منابع

Chakrabarty, K., Chandra., 2001. System-on-a-chip test data compression and decompression architectures based on golomb codes, IEEE Trans. Computer Aided Design, Vol. 20, pp. 355-368.

Chandra., Chakrabarty, K., 2003. Test data compression and test resource partitioning for system on a chip using frequency-directed run-length (FDR) codes, IEEE Transactions on computer. Vol. 52.

Chandra., Chakrabarty, K., 2000. Test data compression for system-on-a-chip using golomb codes, Proc. IEEE VLSI Test Symp, pp. 113-120.

Cobourn, W.G., Dolcine, L., French, M., Hubbard, M.C., 2009. A Comparison of Nonlinear Regression and Neural Network Models for Ground-Level Ozone Forecasting, J. Air & Waste Manage.

Fukunaga, K., Hummels, D.M., 1989. Leave-one-out procedure for nonparametric error estimate, IEEE transactions on pattern analysis and machine intelligence, Vol. 11, pp. 421-423.



پوریا اعتزادی‌فر در سال ۱۳۶۸ در شهرستان نیشابور به دنیا آمد. دوره کارشناسی خود را در دانشگاه بیرجند در رشته برق - مخابرات در سال ۱۳۹۰ به پایان رسانید. او در حال حاضر دانشجوی کارشناسی ارشد مهندسی برق - مخابرات در دانشگاه بیرجند است. زمینه تحقیقاتی ایشان پردازش سیگنال‌های دیجیتال و تصویر است.

نشانی رایانامه‌های ایشان عبارت است از:

P.etezadifar@birjand.ac.ir
P.etezadifar@gmail.com

Shi, Y.Q., Sun, H., 2000. Image and Video Compression for Multimedia Engineering Fundamentals, Algorithms and Standards, CRC Press.

Stone, M., 1974. Cross-validators choice and 1 assessment of statistical predictors, Journal of the Royal Statistical Society, Vol. 36, pp. 111-147.

Touba, N.A., Jas, A., 1998. Test vector decompression via cyclical scan chains and its application to testing core-based design, Proc. Int'l Test Conf, pp. 458- 464.

Univ. of Illinois IGATE website www.crh-c.uiuc.edu/IGATE.2000.

Vemuri, V., 1988. Artificial Neural Networks, Theoretical Concepts, Computer Society Press Technology Series, Computer Society of Institute of Electrical and Electronics Engineers, Washington, DC.

حسن فرسی در سال ۱۳۴۸ در شهرستان بیرجند به دنیا آمد. دوره کارشناسی و کارشناسی ارشد خود را در رشته مهندسی

برق - مخابرات در دانشگاه صنعتی

شریف - تهران به ترتیب در سال‌های

۱۳۷۱ و ۱۳۷۴ به پایان رسانید. از سال

۱۳۷۱ به مدت یک‌سال در مرکز تحقیقات

مخابرات ایران اشتغال داشت. در سال

۱۳۷۴ به عضویت هیئت علمی دانشگاه بیرجند درآمد و مدرک دکترای خود را از دانشگاه Surrey انگلستان در زمینه پردازش سیگنال‌های دیجیتال اخذ کرد. در حال حاضر وی عضو هیئت علمی دانشگاه بیرجند است. زمینه کاری ایشان پردازش سیگنال‌های دیجیتال، پردازش صحت و تصویر هستند.

نشانی رایانامه ایشان عبارت است از:

hfarsi@birjand.ac.ir

